REMARKS/ARGUMENTS

Claims 1-26 are pending in the present application. Claims 10 and 20 are canceled. Claims 1, 2, 4, 7-9, 11, 12, 18, 19, 21, 22, and 24-26 are amended. Support for the amendments can be found on page 16 through page 24 of the Application as filed. Reconsideration of the claims is respectfully requested.

I. Interview Summary

On Wednesday, February 7, 2007, an Examiner interview was conducted between the Examiner and Mr. Brandon Williams. Amendments to claim 1 were discussed. Additional amendments have been made. No agreement has been reached with regard to the additional amendments.

II. 35 U.S.C. § 101

The examiner rejected claims 21-25 are rejected because the claimed invention is directed to non-statutory subject matter. Applicants have amended claim 21 accordingly, thereby overcoming the rejection.

III. 35 U.S.C. § 102, Anticipation

The examiner rejected Claims 1, 8-11, 18-21 and 26 as anticipated by *Kauffman et al.* Method and Apparatus for Communication in a Multi-Processor Computer System, U.S. Patent 6,332,180 (December 18, 2001) (hereinafter "*Kauffman*"). This rejection is respectfully traversed. Claim 1 is representative of claims 11, 21, and 26. The examiner states that:

> **With respect to claim 1**, Kauffman discloses a method in a data processing system for generating a notification of an event (abstract and column 29, lines 28-36), the method comprising:
>
> responsive to detecting a presence of the event in a platform in the data processing system, generating a hardware interrupt to an operating system (column 29, line 58-column 30, line 12 and column 31, lines 19-21 - source instance generates inter-processor interrupt);
>
> responsive to the presence of the event, storing the event in a partition queue associated with a partition firmware (column 29, line 58-column 30, line 12 and column 30, line 53-column 31, line 1 - source instance modifies appropriate event bit in bitvector or copies relevant event data to packet queue of target);
>
> responsive to receiving a request to check the hardware interrupt in the partition firmware, identifying ,the event in the partition queue (column 30, lines 13-30 and column 31, lines 21-38 - target instance examines event data); and

> responsive to identifying the event, processing the event (column 30, lines 30-38 and column 31, lines 38-43 - target instance calls appropriate processing routine).

Office Action dated November 13, 2006, pp. 2-5.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). In this case each and every feature of the presently claimed invention is not identically shown in the cited reference, arranged as they are in the claims.

Claim 1 as amended is as follows:

> 1. A method in a data processing system having at least two partitions, a platform, partition firmware, and abstracted hardware owned by the platform and not owned by the at least two partitions, wherein the method is for generating a notification of an event caused by the abstracted hardware, the method comprising:
>
> responsive to detecting the event caused by the abstracted hardware in the platform, storing, by the platform, the event in a queue associated with the partition firmware before an interrupt is issued;
>
> generating, by the platform, an external hardware interrupt for the event, wherein the platform sends the external hardware interrupt to an operating system operating on a first partition of the at least two partitions;
>
> responsive to receiving, by the partition firmware, a request from the operating system to read the external hardware interrupt, providing a source of the event to the operating system;
>
> responsive to receiving, by the partition firmware, a request from the operating system to query the queue for the external hardware interrupt identifying, by the partition firmware, the event in the queue; and
>
> responsive to identifying the event, processing the event.

Kauffman does not anticipate claim 1 as amended because *Kauffman* does not teach the claimed feature of, "responsive to detecting the event caused by the abstracted hardware in the platform, storing, by the platform, the event in a queue associated with the partition firmware before an interrupt is issued," as in claim 1. Instead, *Kauffman* teaches that information is communicated from one operating system to another operating system running on another partition. A single bit communication scheme utilizes the shared memory between the two partitions to alert other operating systems through an interprocessor interrupt, as taught by the following portion of *Kauffman:*

> Numerous situations arise during system operation that require information to be conveyed from one operating system to an operating system running on another

partition. In the present invention, there are two preferred ways to communicate between operating system instances. The first is through "single-bit communication," and the second is through the transmission of packetized data. Each of these communication types makes use of the system shared memory and is described in more detail below.

Single bit communication takes advantage of the fact that within the shared memory structure of the present invention are sets of bits used for notification of predetermined system events. For each partition, one set of notification bits (referred to herein as the "notification bitvector") is allocated from the shared memory. In the preferred embodiment, the bits of each notification bitvector are normally set to zero. The instance running on a particular partition, when receiving an interprocessor interrupt (IPINT), examines its notification bitvector and, if any of the bits are nonzero, the instance takes an action according to the bit that has been set. Since each bit of the bitvector represents a different event, identification of the modified bit provides notification of the corresponding event. This allows an instance running on one partition (the "source" instance) to signal an event to an instance running on another partition (the "target" instance) by modifying the appropriate bit of the bitvector associated with the target instance, and then sending the primary CPU of the target instance an IPINT to cause it to examine its notification bitvector. An overview of the steps involved in this type of communication is shown in FIGS. 9A and 9B.

Kauffman, col. 29, ll. 26-56.

Additionally, *Kauffman* teaches a second method of communication between operating system instances by using data packets. *Kaufmann* describes this second communication method as follows:

The second form of communication within the system uses the transmission of data packets. Since some inter-instance communications require not just the signaling of a predetermined event, but the transmission of specific data as well, a mechanism is provided for allowing such communication. A source instance makes use of shared memory space for storing the data to be transmitted, which is thereafter located and copied by the target instance. A general overview of the details of this packet communication process are shown in the flowchart of FIGS. 10A and 10B.

The process starts in step 1000, and then proceeds to step 1002, in which the source instance (i.e. the instance having data to communicate) determines the amount of memory space necessary for storing the data to be transmitted. This memory space is then allocated by the source instance within the shared memory of the APMP database (step 1004), and the data is copied to the allocated memory space (step 1006). After copying the data, the source instance determines a local virtual address for the listhead of the packet queue of the target instance (step 1008). This is accomplished in the same manner as described above for determining the local virtual address for the notification bitvector. The source instance then places the packet in the queue listhead (step 1009).

The packet processing of the target instance may be in any known manner, and is the details of such are not germane to the present invention. However, in this

> embodiment, the target instance has a packet queue with a "listhead" containing the relative offset of the first packet to be processed. The listhead is stored in shared memory, allowing other instances to place packets in the queue in chronological or sequential order. The packet processing in this embodiment is based on relative addressing from one packet to the next to cover the multiple virtual address issue between instances. However, for the purposes of understanding the present invention, it is sufficient to recognize that providing the packet to the queue listhead (i.e., identifying the packet to the listhead, including providing its location in shared memory) is enough to allow the packet processing of the target instance to remove the packet based on the common listhead and make it available to its processing routine.

Kauffman, col. 30, l. 39 through col. 31, l. 13.

However, in neither form of communication described in *Kauffman,* interprocessor interrupts are not generated for an event caused by abstracted hardware, as required by amended claim 1. The specification defines abstracted hardware as follows:

> In this manner, the mechanism of the present invention provides a way to relay platform specific information to a partition from hardware that is abstracted from the partition. Abstracted hardware, as used herein, is any hardware that the system cannot understand. In other words, abstracted hardware is any resource owned by the platform, but is not owned by the partition. For example, an operating system does not understand events regarding power or temperature because no device drivers are present for the sensors that detect these parameters in the platform. These sensors are examples of abstracted hardware.

Specification, p. 16, ll. 4-15 (emphasis supplied).

In *Kauffman*, the system, including all operating system instances, understands all hardware. Thus, *Kauffman* does not teach the claimed feature of "abstracted hardware," as arranged in claim 1. Accordingly, *Kauffman* does not teach this claimed feature.

Stated differently, the interprocessor interrupts of *Kauffman* are, by definition, excluded from the claim language of amended claim 1. Each partition in *Kauffman* necessarily owns a processor. Processor interrupts are therefore never generated for an event caused by abstracted hardware, required in amended claim 1, because abstracted hardware is not understood by the system. Accordingly, *Kauffman* does not teach the claimed feature of, "responsive to detecting the event caused by the abstracted hardware in the platform, storing, by the platform, the event in a queue associated with the partition firmware before an interrupt is issued," as in amended claim 1.

For similar reasons, *Kauffman* also does not teach the claimed feature of, "generating, by the platform, an external hardware interrupt for the event, wherein the platform sends the external hardware interrupt to an operating system operating on a first partition of the at least two partitions," as in amended claim 1. For similar reasons, *Kauffman* also does not teach the claimed feature of, "responsive to receiving, by the partition firmware, a request from the operating system to read the external hardware

interrupt, providing a source of the event to the operating system," as in amended claim 1. Likewise, *Kaufmann* also does not teach the claimed feature of, "responsive to receiving, by the partition firmware, a request from the operating system to query the queue for the external hardware interrupt identifying, by the partition firmware, the event in the queue," as in amended claim 1.

As shown above, *Kauffman* does not teach all of the features of claim 1 as amended. Accordingly, under the standards of *In re Lowry*, *Kauffman* does not anticipate claim 1 as amended. Additionally, the remaining independent claims all contain features similar to those presented in claim 1. Therefore *Kaufmann* does not anticipate the independent claims for the reasons stated above.

Because claims 8-10 depend from claim 1 and claims 18-20 depend from claim 11, the same distinctions between *Kauffman* and the claimed invention in claims 1 and 11 applies to each of these dependent claims. Therefore, the rejection of claims 1, 8-11, 18-21 and 26 has been overcome.

IV. 35 U.S.C. § 103, Obviousness

The examiner rejected Claims 2-7, 12-17, and 22-25 as obvious over *Kauffman* in view of *Ahrens et al.*, Standardized Format for Reporting Error Events Occurring within Logically Partitioned Multiprocessing Systems, U.S. Patent 6,792,564 (September 14, 2004) (hereinafter *Ahrens*). This rejection is respectfully traversed.

The examiner states that:

> **With respect to claim 2**, Kauffman does not disclose expressly wherein the processing step includes initiating a corrective action; and sending a notification to the operating system.
>
> Ahrens teaches a method for reporting error events in a logically partitioned multiprocessing system (abstract and column 1, lines 9-15), which includes initiating a corrective action (column 5, lines 50-58); and sending a notification to the operating system (column 7, lines 9-14).
>
> At the time the invention was made, it would have been obvious to a person of ordinary skill in the art to modify Kauffman by initiating a corrective action and sending a notification to the operating system, as taught by Ahrens. A person of ordinary skill in the art would have been motivated to do so because, although Kauffman discloses processing of events (column 30, lines 30-38 and column 31, lines 38-43), they are not restricted to error events which need to be corrected. However, since Kauffman does disclose taking action depending upon the processed event (column 29, lines 40-56), it would have been highly desirable to incorporate the teachings of Ahrens, to initiate corrective actions and send a notification to the operating system, in the event of an error.

Office Action dated November 13, 2006, pp. 5-7.

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). A *prima facie* case of obviousness is established when the teachings of the prior art itself

suggest the claimed subject matter to a person of ordinary skill in the art. *In re Bell*, 991 F.2d 781, 783, 26 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1993). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). For an invention to be *prima facie* obvious, the prior art must teach or suggest all claim limitations. *In re Royka,* 490 F.2d 981, 180 USPQ 580 (CCPA 1974). In the case at hand, the cited references do not teach or suggest all of the features of the claims, arranged as they are in the claims.

Ahrens teaches methods and systems for reporting errors within a computer system. When an error event occurring within one logical partition is detected, information about the error event is formatted according to a specified format. Each operating system utilizes this certain format for error reporting. *Ahrens*, col. 1, ll. 58-67.

Ahrens does not teach or disclose the deficiencies of *Kauffman* now required by amended claims 1, 11, and 21, namely "responsive to receiving, by the partition firmware, a request from the operating system to query the queue for the external hardware interrupt identifying, by the partition firmware, the event in the queue." Therefore, the combination of *Kauffman* and *Ahrens*, when considered as a whole, does not teach or suggest each feature of amended claims 1, 11, and 21. Consequently, under the standards of *In re Royka*, no *prima facie* obviousness rejection can be made against claims 1, 11, and 21 using a combination of *Kauffman* and *Ahrens*.

Because claims 2-7, 12-17, and 22-25 depend from claims 1, 11, and 21 respectively, no *prima facie* obviousness rejection can be made against these claims for similar reasons. Consequently, the rejection of claims 2-7, 12-17, and 22-25 has been overcome.

V. Conclusion

The subject application is patentable over the references and should now be in condition for allowance. The examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: February 13, 2007

Respectfully submitted,

/Theodore D. Fay III/

Theodore D. Fay III
Reg. No. 48,504
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants

TF/BW